Srikar Arani | Perm: 3223005 | srikar.arani@gmail.com

Professor Xifeng Yan

CS 165A

6 March 2021

Machine Project 2: Pac Man Search

Architecture

The Pac Man program was written using python. The only edited file was the file, "multiAgents.py" which consists of the edited class, "MultiPacmanAgent" and the custom evaluator function "myEvaluationFunction".

Model Building

The class "MultiPacmanAgent" consists of two functions: "getAction" and "minimax". The "getAction" function initializes all the variables needed for the minimax function, and then calls the function and returns the result of the minimax tree. It sets the alpha and beta to negative and positive infinity respectively. It then loops through all legal actions and runs a minimax tree to find the ideal move.

The minimax tree itself implements a recursive function strongly based on the alpha-beta implementation image included with the starter code. The code takes as argument, the current game state of the pac man, the depth in the tree with 0 at root, the alpha and beta values, the index of the agent, and a simple boolean for whether or not the maximizer or minimizer is being called. This was done in order to make the code more reusable and less complex. The maximizer follows the provided example closely. The minimizer has to consider whether or not the ghost is the agent. The minimizer takes this into account and maximizes the ghosts chance while minimizing the player's chance.

The evaluator function finds the current score of the agent. Using some of the functions provided with the reflex agent evaluator function, the author calculated the distance to food and ghosts using the manhattan distance. The author took a linear combination of these factors and tried and tested different weight values until a desirable score was reached.

Results

The final program runs as expected. It passes 9 of 10 tests, and has an average score of 1766.1. This is above the minimum cut off of 1500, but not very high overall.

Challenges

The project seemed initially rather complex to even get a handle on. In order to combat this, the author first started by recreating a minimax tree with a much simpler idea. The author also utilized many UC Berkeley lectures that talked about the project and the approaches to the project. This helped explain the topic much more easily. The actual minimax function was rather simple to implement. However, the evaluation function was the very complex part. Even now, it is far from perfect.

Weaknesses

This model is not perfectly calibrated even for the current pac man tests. Like mentioned earlier, the biggest weakness with the minimax algorithm is setting an evaluator to calculate scores. This model has a very simplistic linear combination of factors as an evaluator. A more appropriate evaluator would not randomly guess and check different weights, but actually understand how to analytically arrive at certain values.